Planificación

Gustavo Romero

Arquitectura y Tecnología de Computadores

18 de enero de 2011

Índice

Lecturas recomendadas

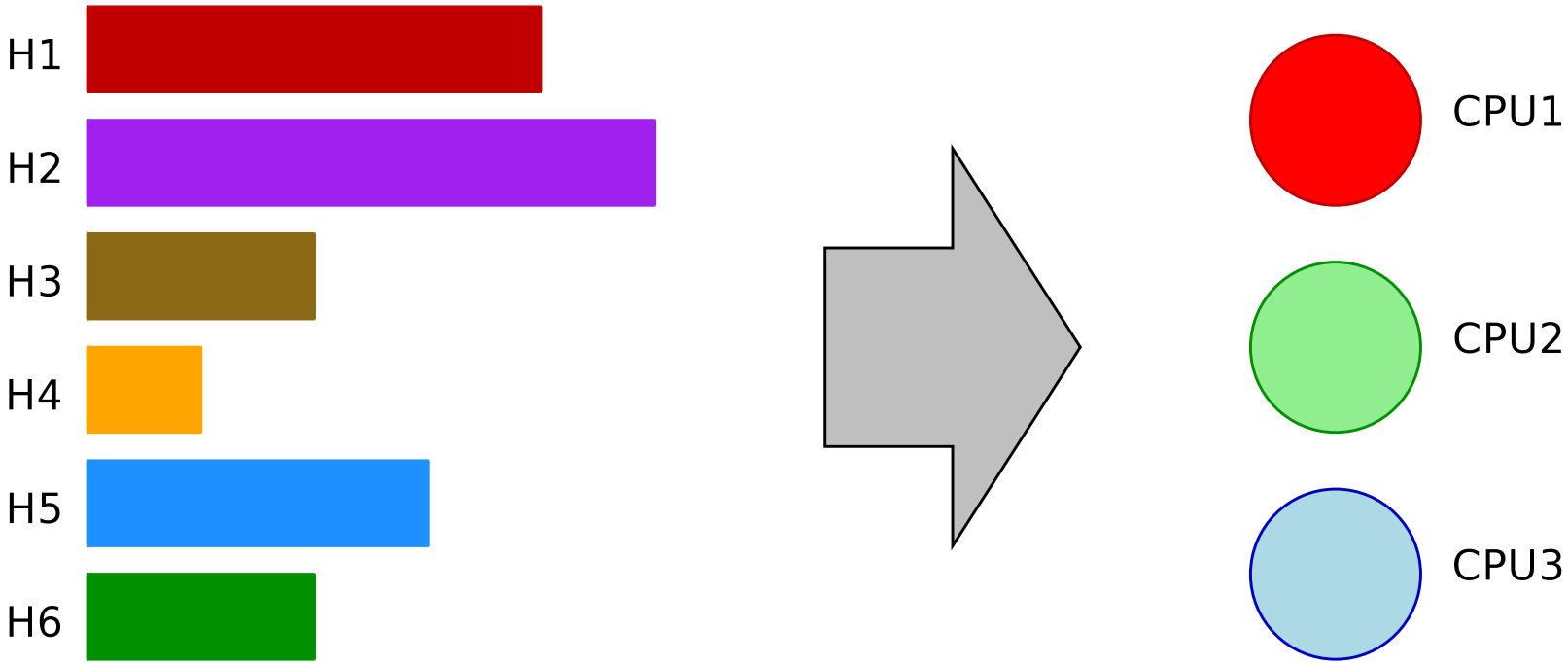
Silberschatz	Fundamentos de Sistemas Operativos (5, 19.5)
Stallings	Sistemas Operativos (9, 10)
Tanenbaum	Sistemas Operativos Modernos (2.5, 7.4, 8.1.4, 8.2.6)

Introducción

Introducción

- Planificación: **gestión** del recurso más valioso de un sistema, el **procesador**.
- Uno de los aspectos más **influyentes** en el **rendimiento** del SO.
- Función: asignar unidades de ejecución al procesador/es.
- Objetivos:
  - Uso eficiente/poca sobrecarga.
  - Equidad/no inanición.
  - Muchos otros en base a fines específicos.
- Investigación:
  - Uniprocesador: abundante investigación, soluciones aceptables.
  - Multiprocesador/multicomputador/multiproceso/multihebra.
  - Tiempo real.
  - Bajo consumo energético.
  - Evitar la inversión de prioridad.

Problema clásico de planificación



- ¿Cómo asignar estas hebras a los procesadores?
- ¿Existe una planificación óptima?
- Hasta que no acordemos que medida de rendimiento emplear no podemos decidir si una planificación es buena o mala.

Problema clásico de planificación

diagrama de Gantt
LJF - Longest Job First

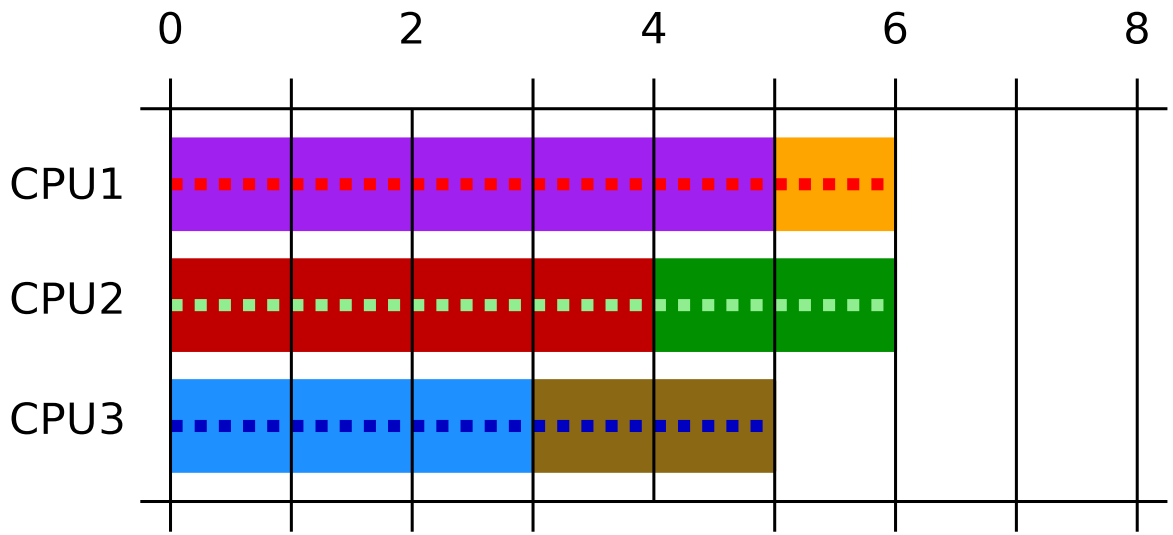
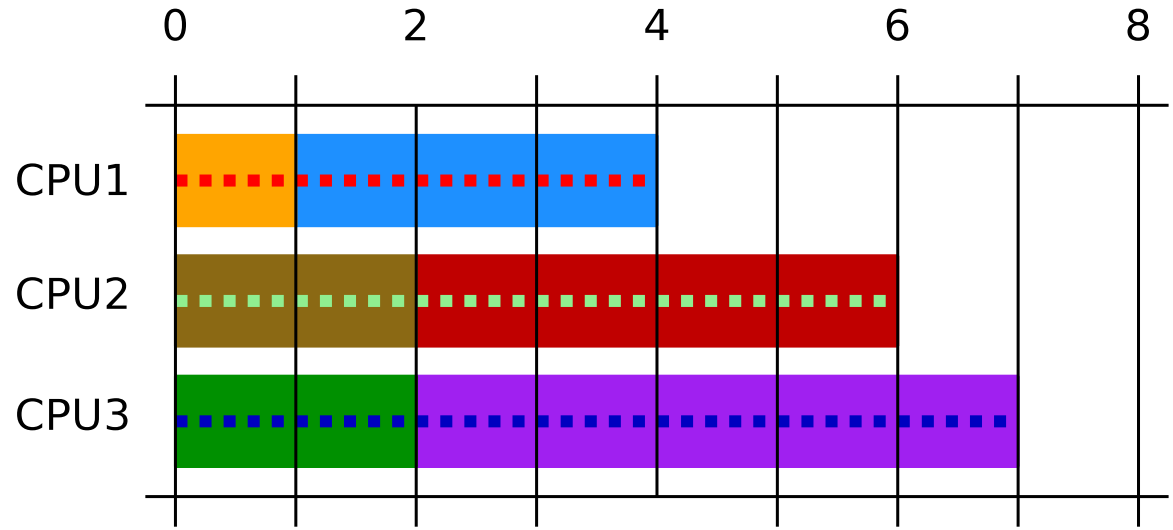


diagrama de Gantt
SJF - Shortest Job First



Parámetros de diseño

- Número de procesadores.
- Conjunto de unidades de planificación estático o dinámico.
- Planificación estática o dinámica.
- Tiempos de ejecución conocidos o desconocidos.
- Expropiación/no expropiación.
- Relaciones de precedencia.
- Uso de Prioridades.
- Cumplimiento de plazos.
- Tipo de sistema: por lotes, interactivo, tiempo real.
- Medidas de rendimiento:
  - Tiempo de respuesta.
  - Tiempo de estancia.
  - Rendimiento.

Objetivos de diseño

- Tiempo de estancia (máximo/medio).
- Tiempo de respuesta (máximo/medio).
- Mínimo número de procesadores.
- Rendimiento: nº de trabajos finalizados por unidad de tiempo.
- Utilización del procesador.
- Cumplimiento de plazos (incumplimiento mínimo/medio).
- Sistema autoadaptativo.

No todos estos objetivos pueden conseguirse simultáneamente con un único algoritmo de planificación.

Principales métricas cuantitativas

Objetivos de rendimiento:

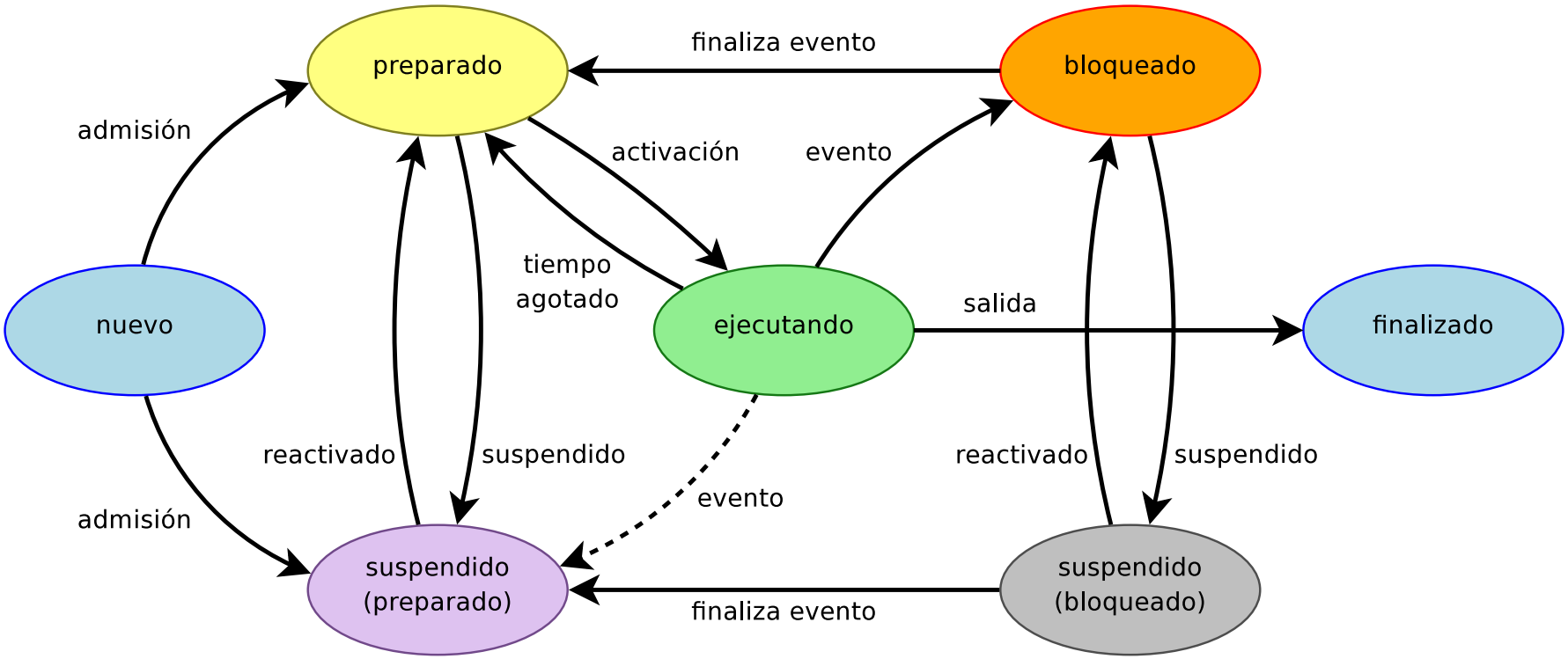
- Alta **utilización** del procesador:
  - Porcentaje de tiempo que el procesador está ocupado.
  - Mediada engañosa en desuso (cierta si libre = desperdiciado).
- Alto **rendimiento**:
  - Número de trabajos completados por unidad de tiempo.
- Bajo tiempo de **estancia**:
  - Tiempo transcurrido desde el **envío** de un trabajo hasta su **finalización**.
- Bajo tiempo de **respuesta**:
  - Tiempo transcurrido desde el **envío** de un trabajo hasta que se **inicia** su ejecución.

Principales métricas cualitativas

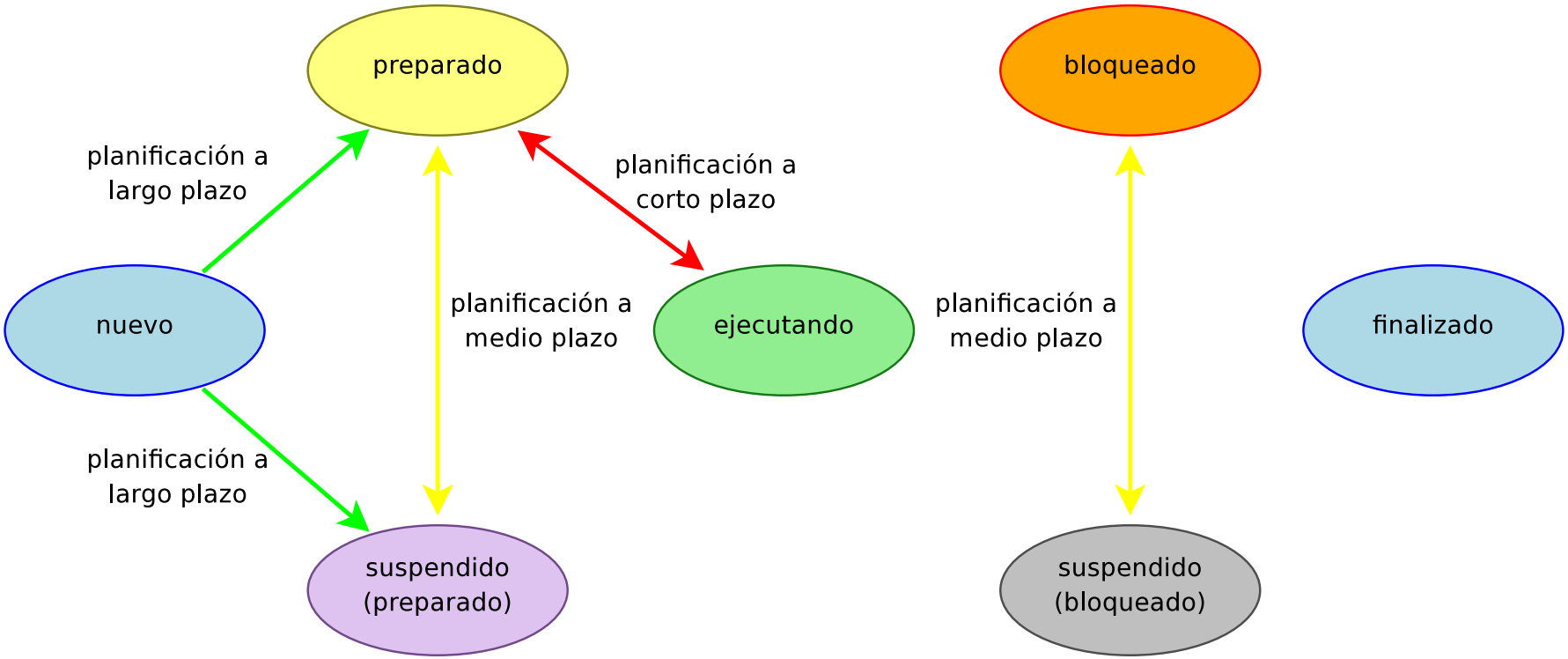
Objetivos del sistema:

- Ningún/escaso incumplimiento de **plazos**:
  - Cumplir los plazos en sistemas de tiempo real.
- Alta **predecibilidad**:
  - Baja **variabilidad** en el tiempo de estancia.
- Alta **robustez**:
  - Ninguna/pocas **caídas** del sistema.
- Alta **equidad**:
  - Ningún/pocos casos de **inanición**.

Estados de un proceso



Tipos de planificación



- largo plazo: que procesos **admitir** al sistema $\longrightarrow$ [s - min].
- medio plazo: que procesos **intercambiar** $\longrightarrow$ [ms - s].
- corto plazo: que proceso **ejecutar** $\longrightarrow$ [μs - ms].

Planificador a largo plazo

- Controla la población de **aplicaciones admitidas para ejecución** en el sistema.
  - Aplicación/tareas del sistema.
  - Grado de multiprogramación.
- **Consecuencias**: si más tareas son admitidas...
  - es menos probable que todas ellas estén bloqueadas esperando por algún evento:

    $\Longrightarrow$ mejor uso del procesador (en algunos casos).
  - cada tarea recibirá una fracción menor de tiempo del procesador:

    $\Longrightarrow$ tiempo de respuesta y estacia más largos.

Planificador a medio plazo

- Toma decisiones sobre **intercambio** para poder controlar el el grado de **multiprogramación**.
- Lo realiza el software de gestión de memoria.
  - Este apartado se verá con más detalle en el tema de gestión de memoria.
  - Asignación del conjunto residente.
  - Control de carga.

Planificador a corto plazo (1)

- Decide cuál es la próxima hebra a ejecutar.
- También se le conoce como **activador** (*"dispatcher"*).
- Es ejecutado cada vez que se produce un evento que puede conducir a un cambio de hebra:
  - interrupciones: reloj, e/s, ...
  - excepciones.
  - llamadas al sistema.
  - señales.
  - ...

Planificador a corto plazo (2)

Criterios orientados al usuario:

Tiempo de **respuesta** Tiempo transcurrido desde el **envío** de un trabajo hasta que **inicia** su ejecución.

Tiempo de **estancia** Tiempo transcurrido desde el **envío** de un trabajo hasta que **finaliza**.

Criterios orientados al sistema:

Utilización del procesador Aprovechamiento frente a desuso (no siempre más uso = mejor aprovechamiento).

Equidad Cada usuario, grupo y aplicación debería disponer del mismo tiempo de procesador.

Rendimiento Número de trabajos completados por unidad de tiempo.

Planificación en sistemas interactivos

- Minimizar el **tiempo de respuesta medio**:
  - Tiempo transcurrido entre la espera (bloqueado/preparado) y la siguiente E/S $\rightarrow$ mejora el uso de los dispositivos de E/S.
  - Iniciar la respuesta al usuario tan rápido como sea posible.
  - Procesar las entradas tan pronto como se reciban.

- Minimizar la **varianza del tiempo de respuesta**:
  - La predecibilidad a menudo es importante.
  - Suele preferirse una media más elevada pero con menor varianza frente a una media menor $\rightarrow$ percepción de la equidad.

Planificación en servidores

- Maximizar el **rendimiento**:
  - Minimizando la sobrecarga del SO $\leftrightarrow$ minimizando el número de cambios de contexto.
  - Haciendo un uso eficiente del procesador y los dispositivos de E/S.

- Minimizar el **tiempo de espera**:
  - Dar a cada proceso el mismo tiempo de procesador.
  - Puede que se incremente el tiempo medio de respuesta.
  - Caso extremo: ejecución secuencial.

Uso de prioridades

Problemas:

- ¿Quién puede establecer las prioridades?
  - El usuario y/o el sistema.
- Alto valor de prioridad $\leftrightarrow$ ¿**alta** o **baja** prioridad?:
  - Cambia de un sistema a otro: Unix $\downarrow$, Windows $\uparrow$.
- El planificador siempre escoge hebras de alta prioridad antes que otras de baja $\longrightarrow$ las hebras de baja prioridad pueden sufrir de **inanición**.
- Para evitar la inanición las hebras susceptibles de padecerla pueden sufrir cambios en su valor de prioridad de forma **dinámica** de manera que mejore en base a...
  - Su edad (tiempo de llegada).
  - Su historia de ejecución (porcentaje de uso del procesador).

Políticas

Características de una política de planificación

¿Cuándo y cómo ejecutar el planificador?

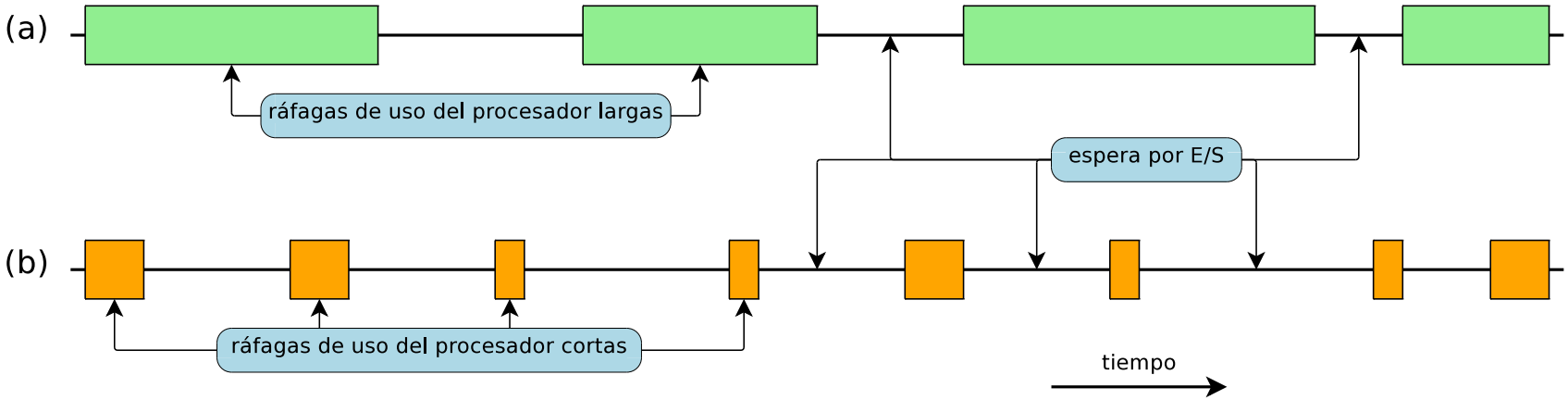
- **Función de selección**: determina que hebra ejecutar a continuación.
  - Esta hebra se escoge de entre las hebras preparadas.

- **Modo de decisión**: especifica los eventos que provocarán la ejecución de la función de selección.
  - **No expulsivo** (*"non preemtive"*): Una vez que una hebra se está ejecutando continua hasta que termina, cede el control voluntariamente o se bloquea (e/s, fallo de página,...).
  - **Expulsivo** (*"preemtive"*): La hebra en ejecución puede ser expulsada, es decir, ser devuelta a la cola de preparadas si...
    - hay trabajos más urgentes y/o prioritarios.
    - ha consumido su quantum de tiempo.

El ciclo procesador - E/S (1)

- Es común que las hebras utilicen el procesador y la E/S síncrona de forma cíclica... ¿Tendría ventajas el uso de E/S asíncrona?
- Cada ráfaga de uso del procesador es seguida de otra de E/S.
- Las ráfagas de E/S suelen ser mucho más largas que las de CPU.
- Toda hebra finaliza su ejecución durante una ráfaga de CPU o bien es abortada por otra durante una ráfaga de E/S o CPU.
- Se denomina hebra limitada (o acotada) por procesador a aquella que requiere ráfagas de uso del procesador más largas que las hebras limitadas por E/S.

El ciclo procesador - E/S (2)

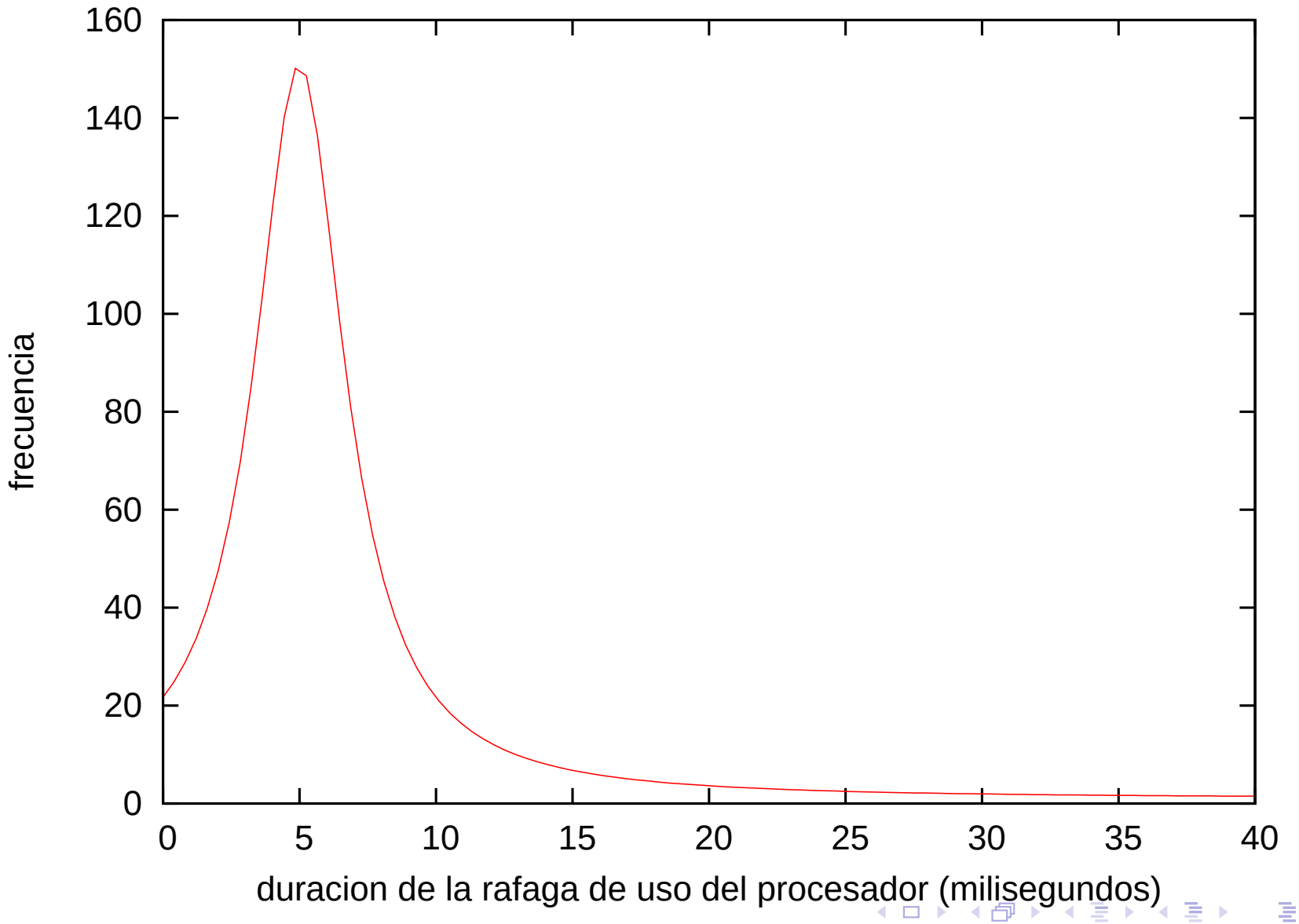


(a) Hebra limitada por CPU.

(b) Hebra limitada por E/S.

Lo que diferencia a los procesos limitados por CPU de los limitados por E/S no es la longitud de las ráfagas de E/S (espera), que son iguales, sino la longitud de las ráfagas de CPU.

Histograma de ráfagas de CPU (Silberschtatz)



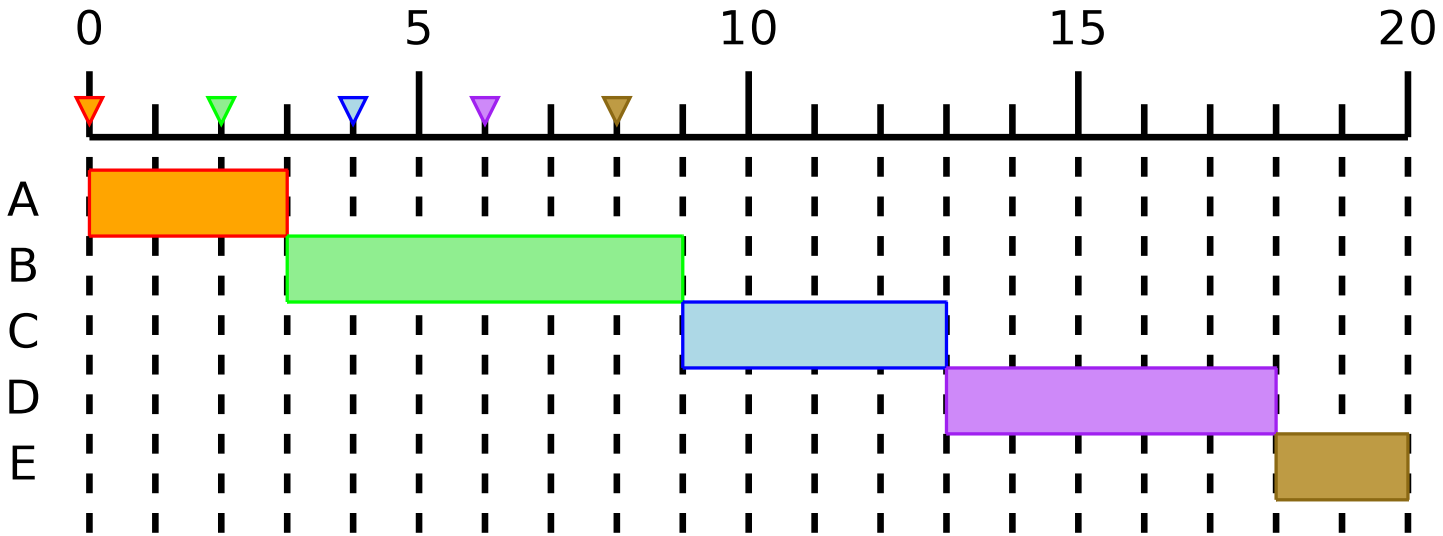
Ejemplo de problema de planificación

trabajo	tiempo de llegada	tiempo de servicio
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

- Tiempo de servicio = tiempo total de procesador requerido.
- Los trabajos con un tiempo de servicio largo (B) están limitados por CPU.
- Utilizaremos este ejemplo para analizar varias políticas de planificación.

Primero en llegar, primero en ser servido

First Come, First Served (FCFS)



- **Función de selección**: escoger la hebra preparada **más antigua**.
- **Modo de decisión**: no expulsivo, una hebra se ejecuta hasta que...
  - coopera (yield),
  - se bloquea (inicia una operación de E/S),
  - provoca una excepción (fallo de página) o
  - finaliza.
- Al igual que en la vida real es un método bastante **justo**, excepto...

¿Cómo implementar FCFS de forma efectiva?

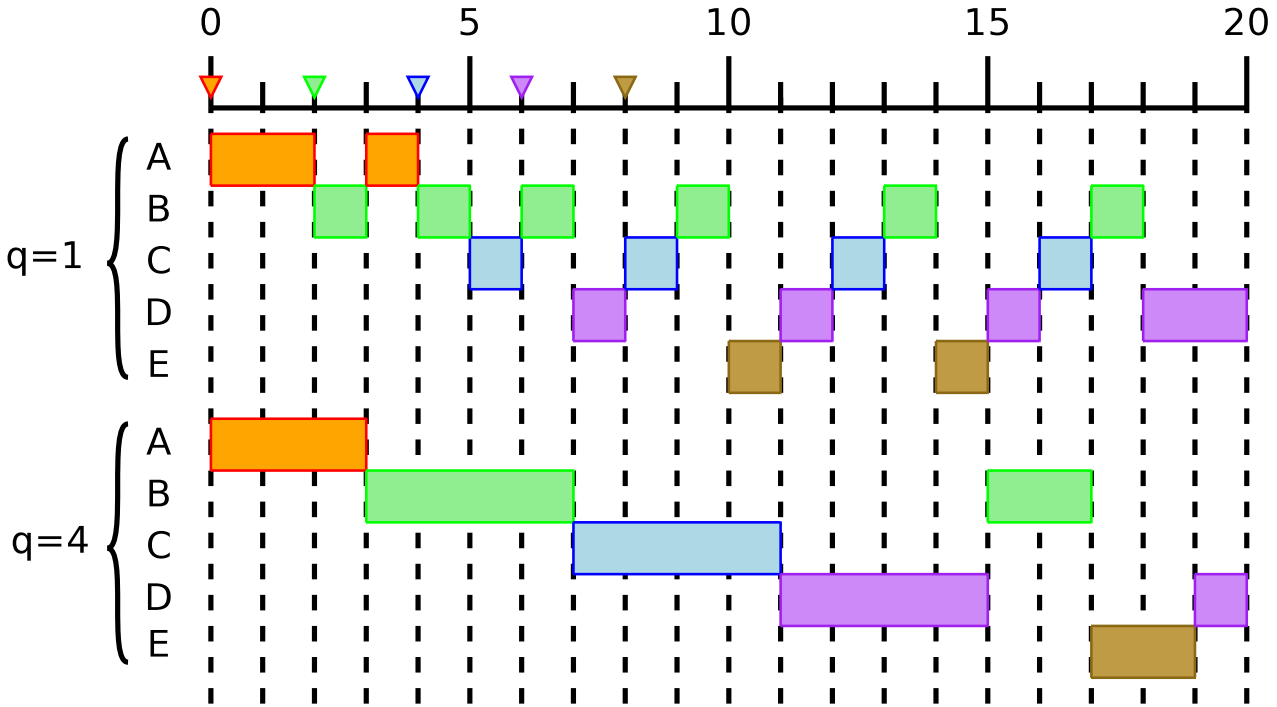
- **Admitir** un nuevo proceso es sencillo: se añade al final de la cola de procesos preparados.
  - Todos los demás procesos deben ser más viejos, y estar por delante, o el control de admisión no funciona bien.
- ¿Qué hacer con un proceso cuando deba ser **desbloqueado**?
  - Añadirlo al final de la cola de preparados no funcionará porque puede que otros procesos más jóvenes o nuevos se hayan adelantado.
- Debemos almacenar la **fecha** de llegada de cada proceso para poder devolverlo a la posición adecuada de la cola de preparados.
  - ¿Es suficiente lo anterior para que sea efectivo?
  - ¿Se le ocurre algún método mejor?

Análisis de FCFS

- **Efecto convoy**: procesos desbloqueados cortos esperan detrás de uno largo.
- Los procesos con poca E/S provocan un **tapón** y **monopolizan** el uso del procesador.
- FCFS favorece a las tareas acotadas por CPU.
  - Las hebras limitadas por E/S deben esperar tras cada operación de E/S hasta que el proceso en ejecución abandone el procesador.
  - **Potencial desperdicio de las capacidades de E/S**.

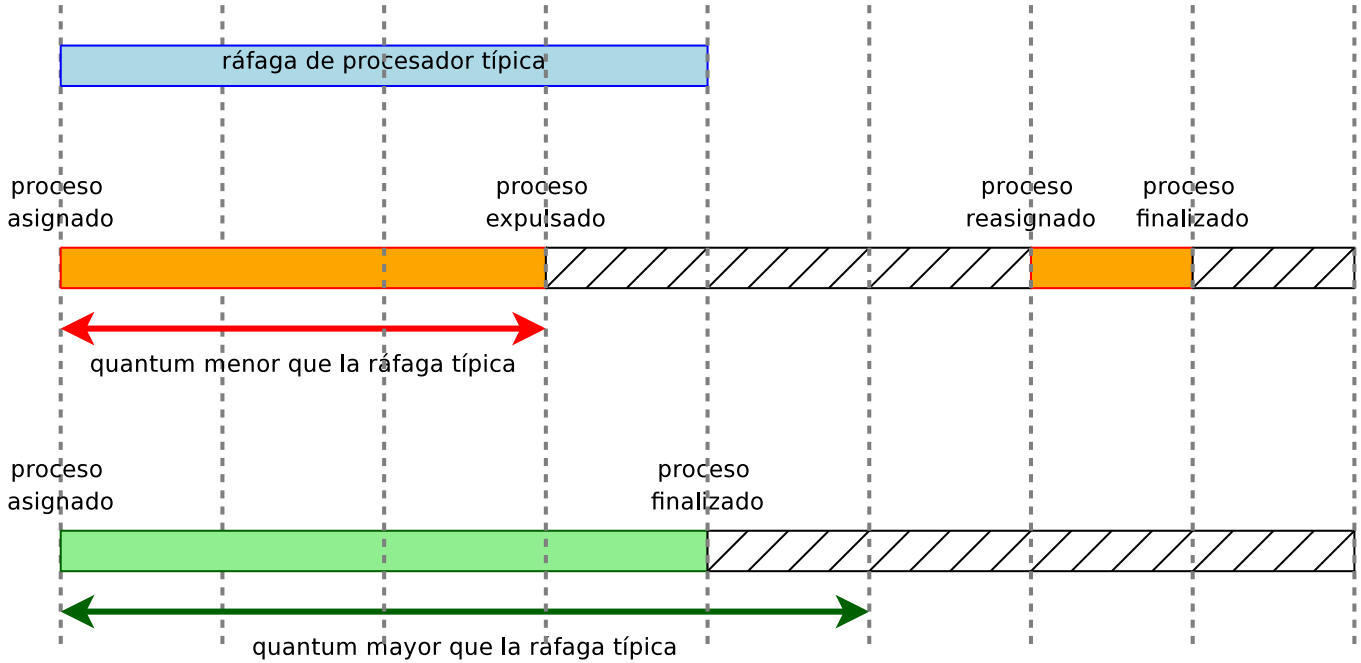
Turno circular

Round Robin (RR)



- **Función de selección**: escoger la **primera** hebra preparada.
- **Modo de decisión**: **expulsivo** por fracción de tiempo (**quantum**).
  - Una hebra se ejecuta hasta **agotar** su quantum ([1, 100]ms).
  - En cada interrupción de reloj se comprueba si se ha agotado el quantum para enviarse al **final** de la cola de **preparados**.

Tamaño del "quantum"



Recomendaciones (en versiones primitivas de Unix quantum = 1s):

- El quantum debe ser mayor que el tiempo de ejecución de la interrupción de reloj y el activador sino sería **ineficiente**.
- El quantum debe ser mayor que la longitud de **interacción típica** para ser **efectivo**, pero no demasiado largo para evitar penalizar a los trabajos limitados por E/S y no acabar imitando a FCFS.

FCFS frente a RR

- Supongamos un cambio de contexto de coste 0, 10 hebras tipo núcleo cada una de las cuales necesita 100s de procesador y que se lanzan todas a la vez y que en RR quantum = 1s.

hebra	Tiempo de respuesta		Tiempo de estancia	
	FCFS	RR	FCFS	RR
0	0	0	100	991
1	100	1	200	992
...	...	...	...	...
8	800	8	900	999
9	900	9	1000	1000
medias	450	4.5	550	994.5

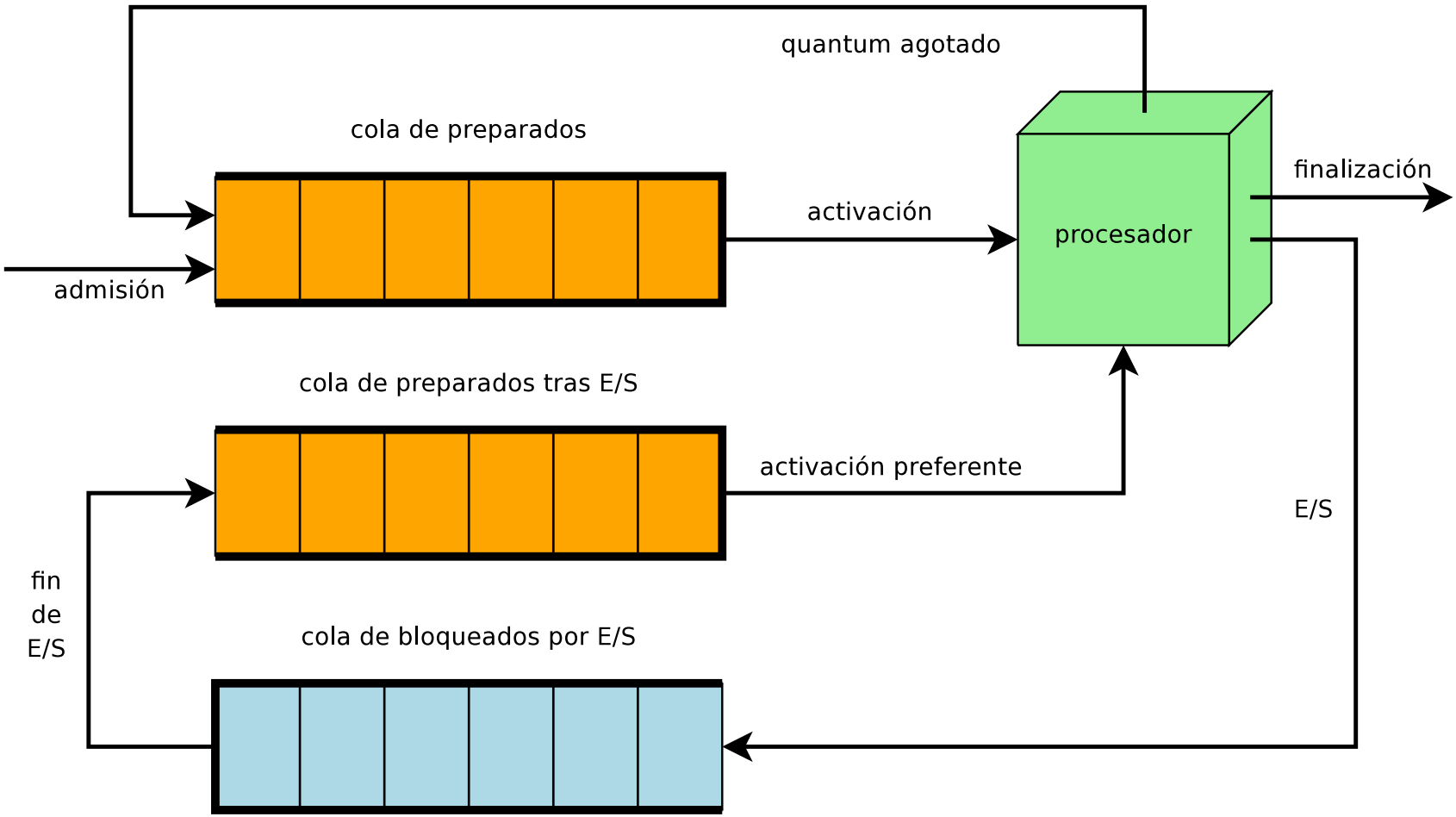
- $\overline{T_r}(FCFS) \ggg \overline{T_r}(RR)$
- $T_e(FCFS) = T_e(RR)$ pero $\overline{T_e}(FCFS) \ll \overline{T_e}(RR)$
- La caché y otros recursos son utilizados por una única hebra en FCFS pero deben repartirse entre muchas en RR.
- En realidad el tiempo total de RR sería mucho mayor (recursos+cambio).

Análisis de turno circular

- **Favorece a las hebras acotadas por CPU**:
  - Las hebras acotadas por E/S no aprovechan su quantum porque se bloquean esperando el fin de la E/S.
  - Las hebras acotadas por CPU además de consumir su quantum completo son devueltas antes a la cola de preparadas.
  - El efecto acumulativo de los dos echos anteriores provoca una gran diferencia de oportunidades.
- Solución: **Turno circular virtual** (*"Virtual Round Robin"* (VRR), Haldar):
  - Cuando la hebra bloqueada en espera de E/S se desbloquea es colocada en otra cola de hebras preparadas de mayor prioridad que la cola de preparadas normal.
  - Las hebras que se ejecutan procedentes de esta cola auxiliar se ejecutan durante el tiempo que dejaron libre de su quantum anterior.

Turno circular virtual

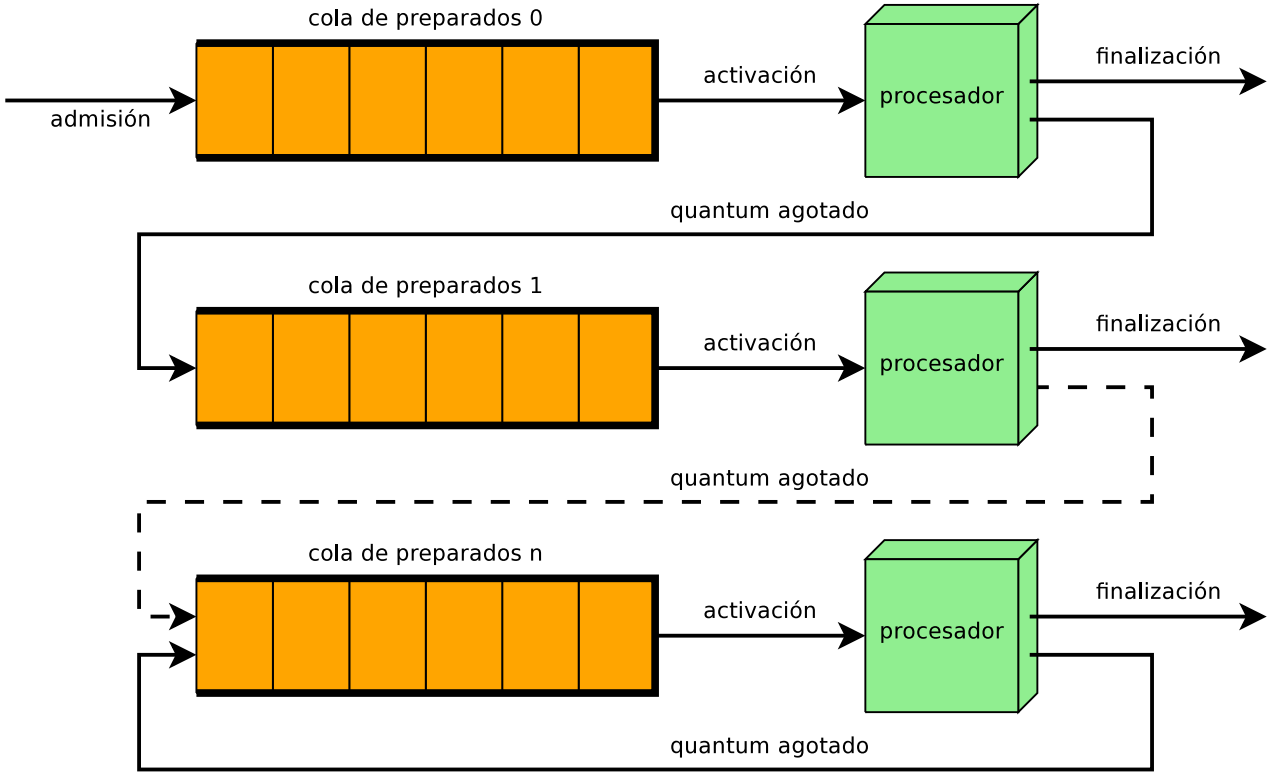
"Virtual Round Robin" (VRR), Haldar



¿Son realmente necesarias dos colas de preparados?

Retroalimentación multinivel

"Multilevel Feedback" (FB)



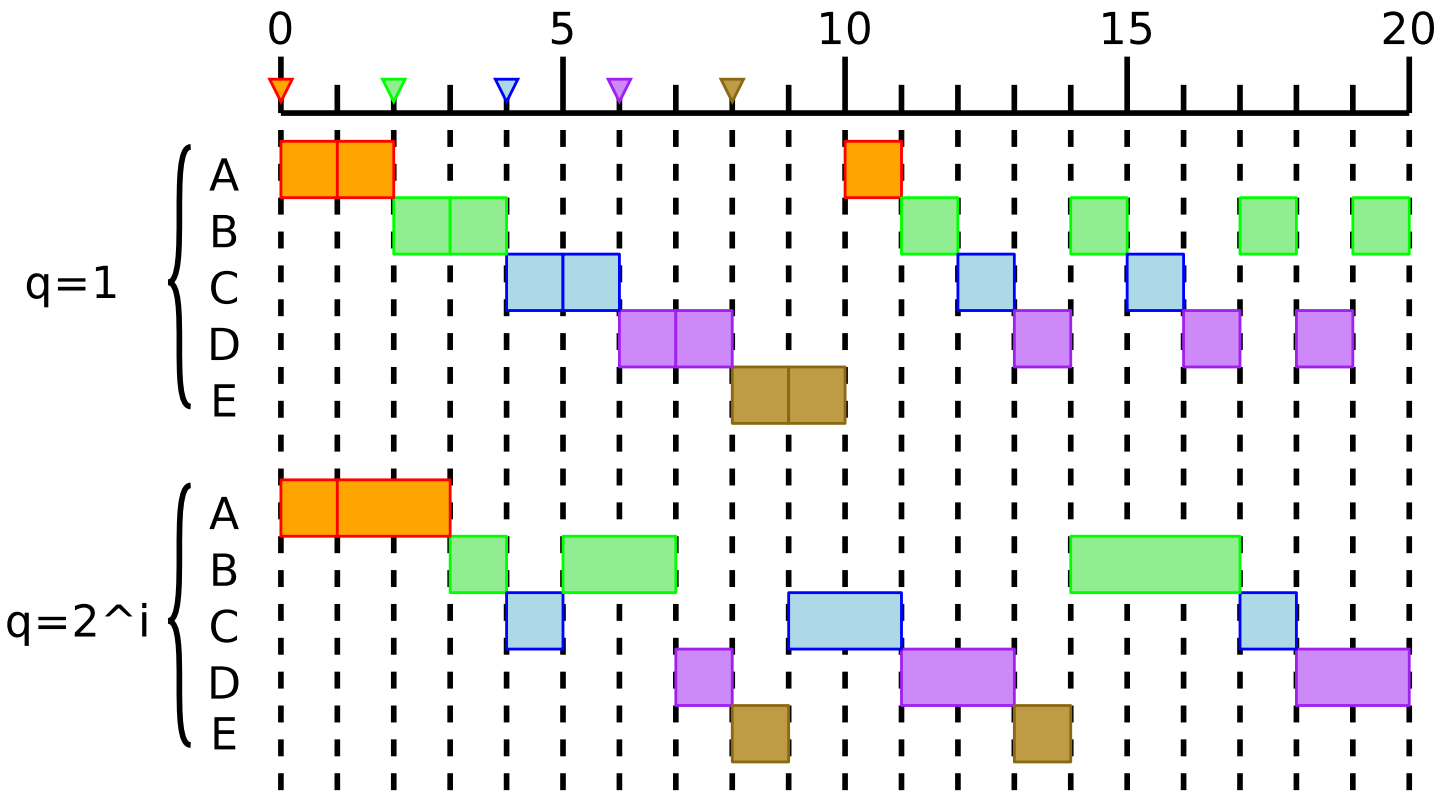
Función de selección **primera hebra** en la **cola de mayor prioridad**.

Modo de decisión **expulsivo** por **quantum** de tiempo, y opcionalmente, por **prioridad**. Cada vez que una hebra agota su quantum es pasada a la cola inferior de menor prioridad.

Análisis de retroalimentación multinivel

- Se parece a "menor tiempo restante primero" (SRTF):
  - Las hebras acotadas por CPU pierden prioridad rápidamente bajando hacia las colas inferiores $\longrightarrow$ posible inanición.
  - Las hebras acotadas por E/S permanecen en las colas de mayor prioridad.
- La planificación se realiza entre colas y dentro de ellas:
  - **Prioridad fija**: seleccionar una hebra de la cola de preparadas i si y sólo si las colas anteriores, $i - 1$ a 0, están vacías.
  - **Quantum de tiempo**: cada cola puede emplear un quantum diferente.
  - La última cola suele planificar mediante FCFS.
- Contramedidas: acciones del usuario para abusar del diseño del sistema, por ejemplo:
  - Añadir operaciones de E/S.
  - Ceder voluntariamente el procesador antes de agotar el quantum.

Quantum en retroalimentación multinivel



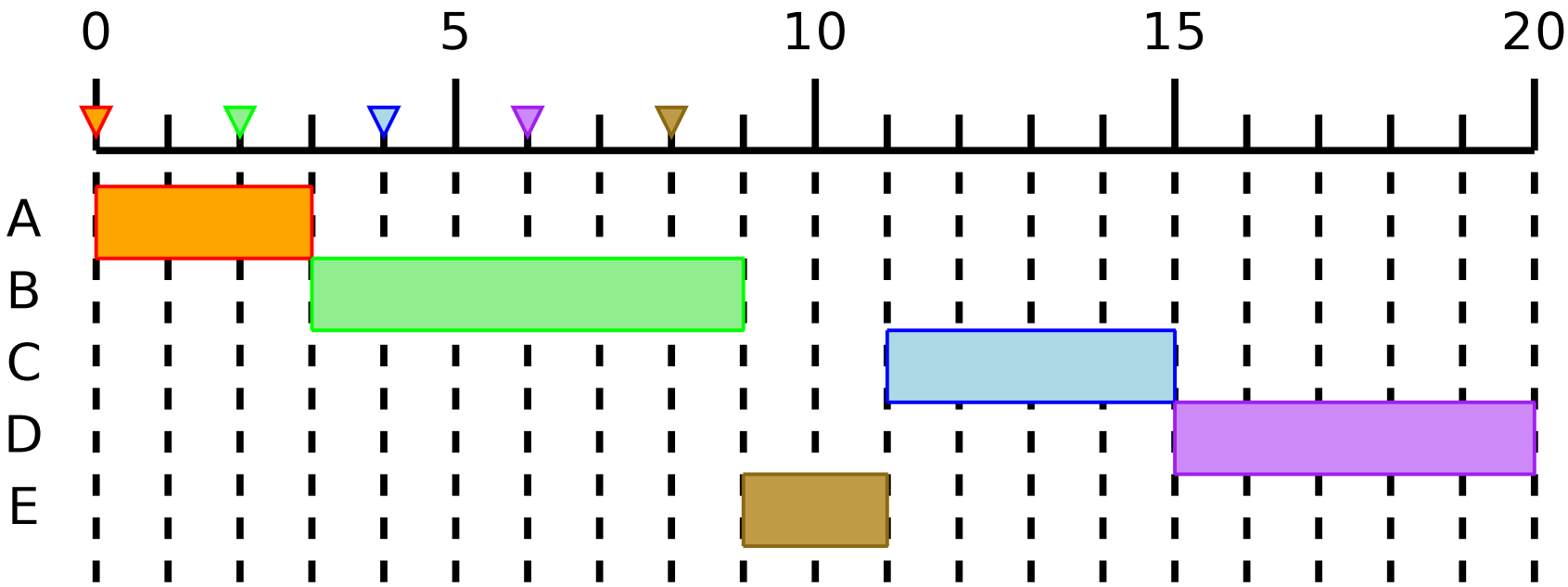
- Un quantum fijo provoca que el tiempo de estancia de las hebras limitadas por CPU se alargue en exceso.
- Para evitarlo se suele incrementar el quantum inversamente a la prioridad de la cola, por ejemplo, $quantum_i = 2^i$

¿Y si conociésemos el futuro?

- ¿Cuánto futuro conocemos o podemos adivinar?
  - La duración completa de un programa.
  - La duración de la siguiente ráfaga de procesador que un proceso va a emplear.
- ¿Ventajas que aportaría?
  - Podríamos finalizar rápidamente los trabajos cortos.
  - Gran efecto sobre trabajos cortos y pequeño sobre los largos.
  - El resultado global es un **mejor tiempo medio de repuesta y de estancia**.

Primero el más corto

"Shortest Job First" (SJF)

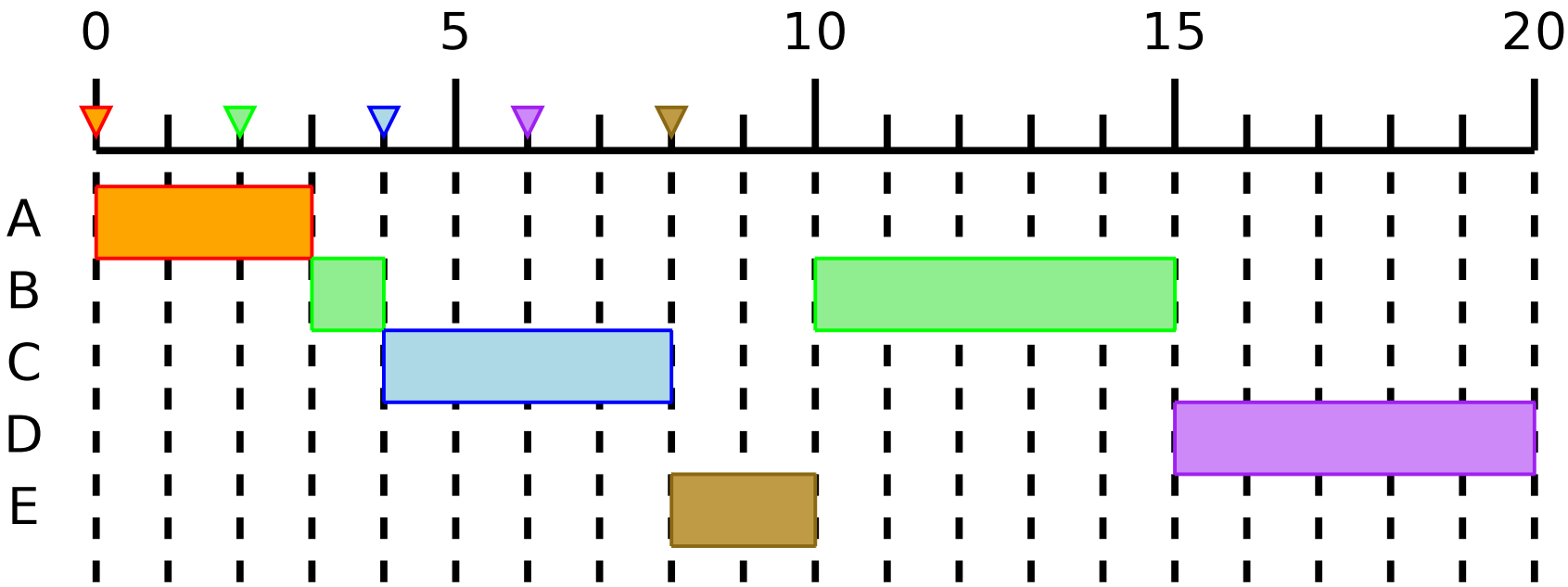


Función de selección: escoger el trabajo que necesite **menor cantidad de CPU**.

Modo de decisión: **no expulsivo**.

Primero el tiempo restante más corto

"Shortest Remaining Time First" (SRTF)



Función de selección: escoger el trabajo al que le **queda menor cantidad de CPU** por usar.

Modo de decisión: **expulsivo**, aplicable en el control de **admisión**.

Estimación de la siguiente ráfaga de CPU (1)

- Sea T_i la duración de la i-ésima ráfaga de CPU de una hebra.
- Llamaremos S_i a la predicción del valor de T_i para la hebra H_i. Una buena estimación haría $S_i = T_i$. La elección más sencilla es:

  $$S_{n+1} = \frac{1}{n}\sum_{i=1}^{n} T_i$$

- Para evitar recalcular toda la suma podríamos modificarlo así:

  $$S_{n+1} = \frac{1}{n} T_n + \frac{n-1}{n} S_n$$

- Esta ecuación considera de igual importancia a todas las ráfagas $\rightarrow$ media aritmética.

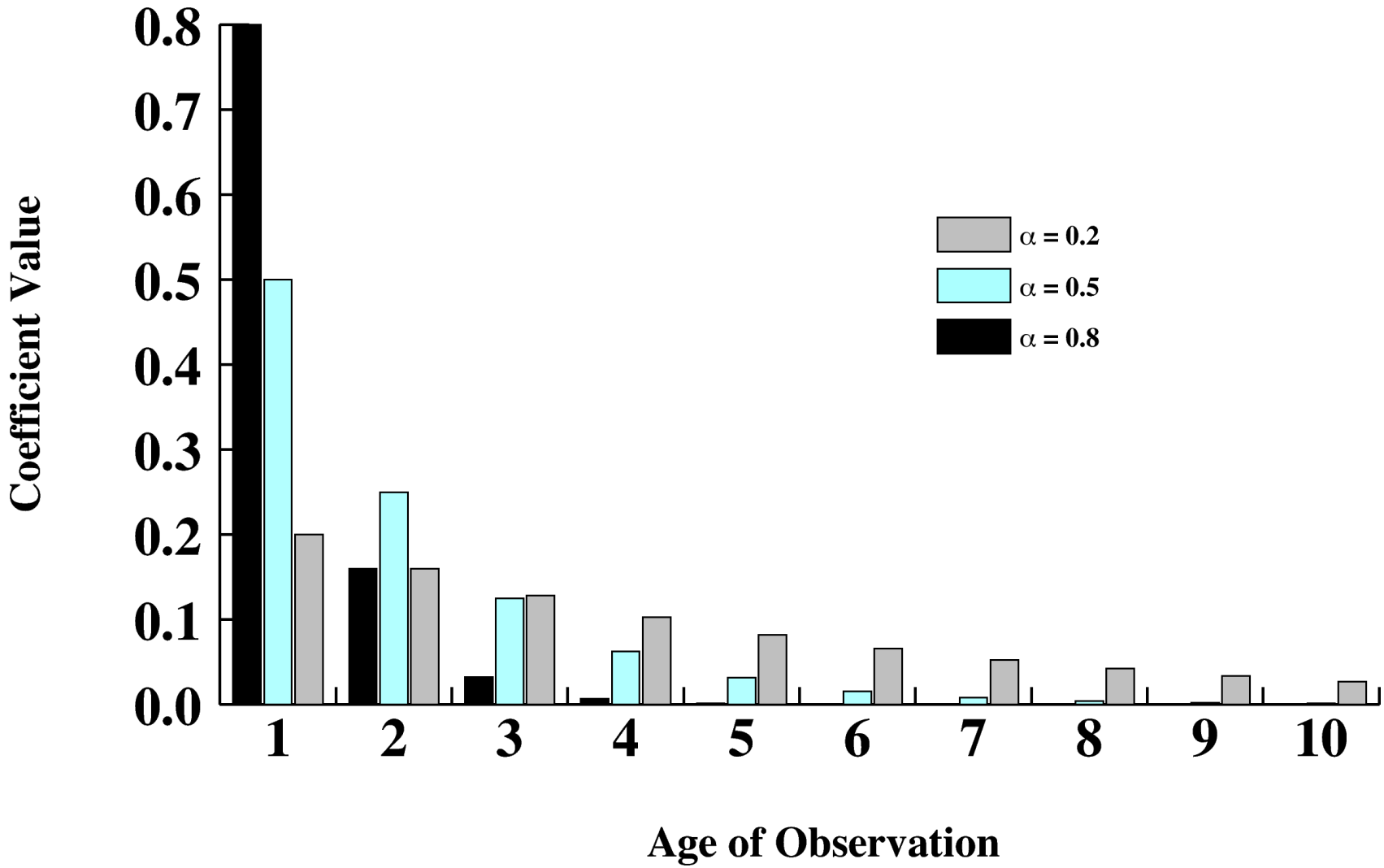
Estimación de la siguiente ráfaga de CPU (2)

- Las ráfagas más recientes representan mejor el comportamiento futuro $\longrightarrow$ media exponencial.

  $S_{n+1} = \alpha T_n + (1 - \alpha) S_n \qquad 0 < \alpha < 1$
- Las ráfagas recientes tienen mayor peso cuando $\alpha > \frac{1}{n}$
- El peso de las ráfagas pasadas decrece exponencialmente:

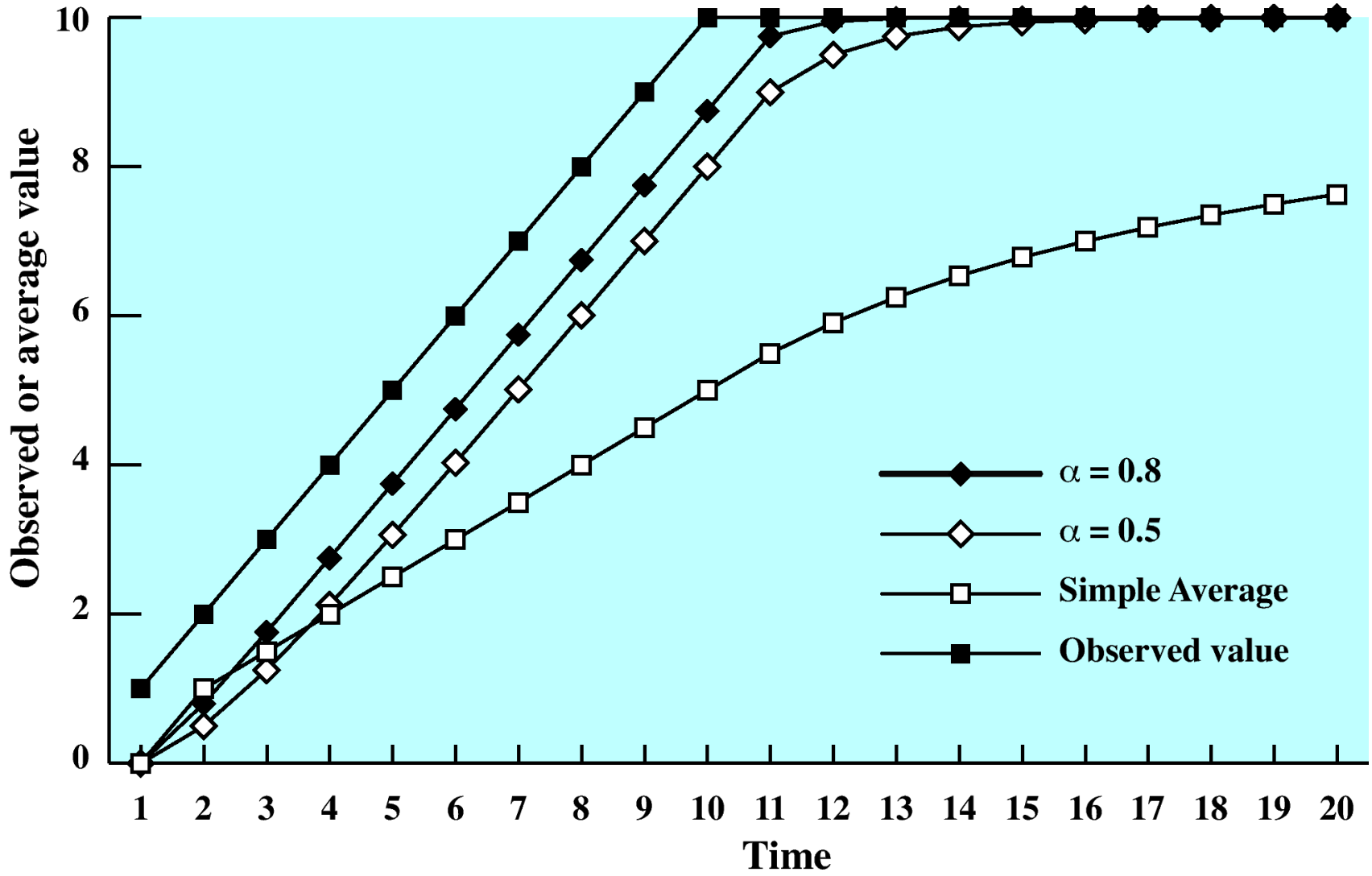
  $S_{n+1} = \alpha T_n + (1-\alpha)\alpha T_{n-1} + \cdots + (1-\alpha)^i \alpha T_{i-1} + \cdots + (1-\alpha)^n \alpha S_1$
- El valor de S_1 no suele calcularse, sino hacerse 0 para asignar la misma alta prioridad a todas las hebras nuevas.
- Un valor frecuente, por eficiencia, es $\alpha = \frac{1}{2}$

Estimación de la siguiente ráfaga de CPU (3)



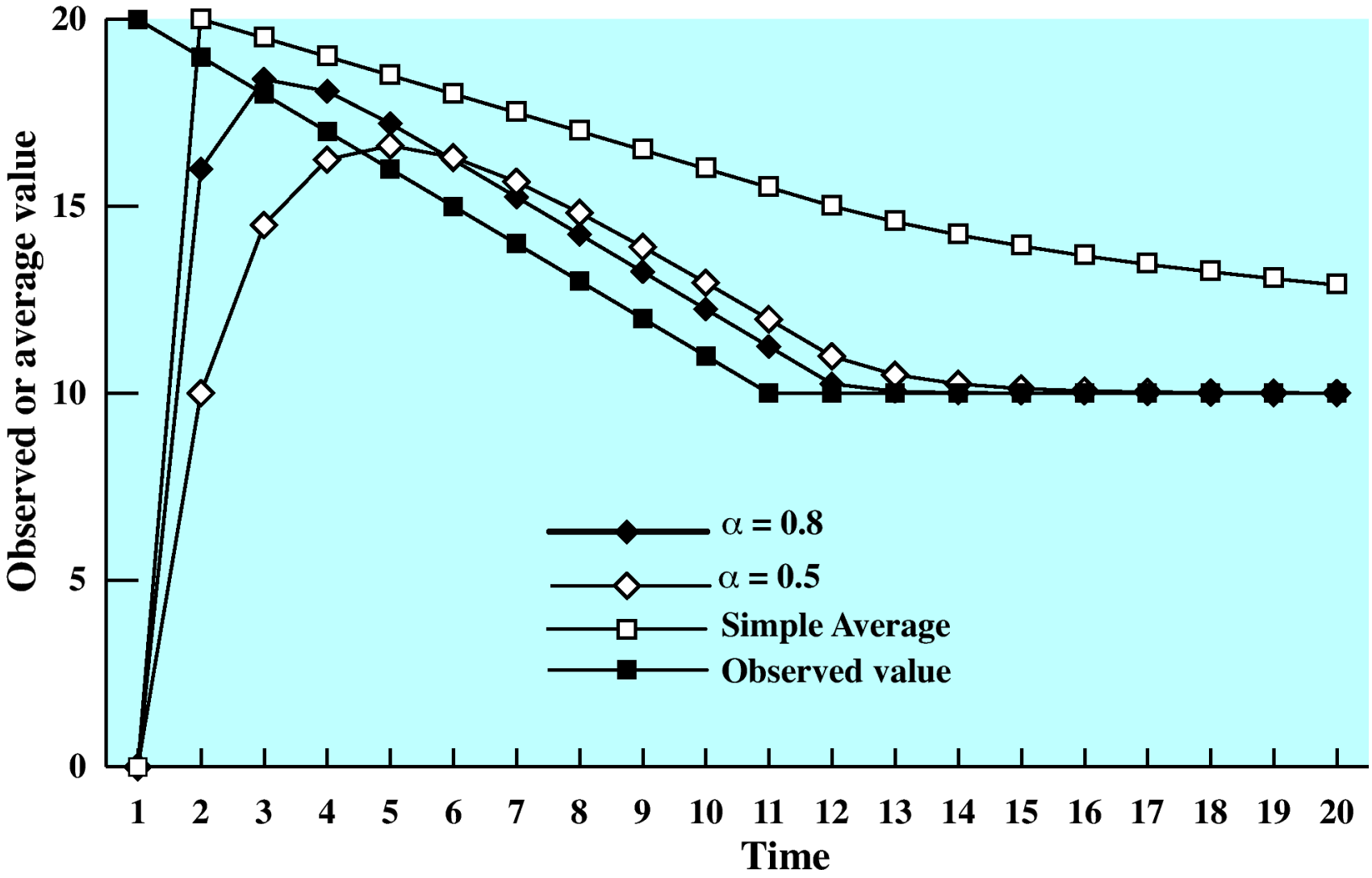
Estimación de la siguiente ráfaga de CPU (4)

incremento del uso de procesador



Estimación de la siguiente ráfaga de CPU (5)

decremento del uso de procesador

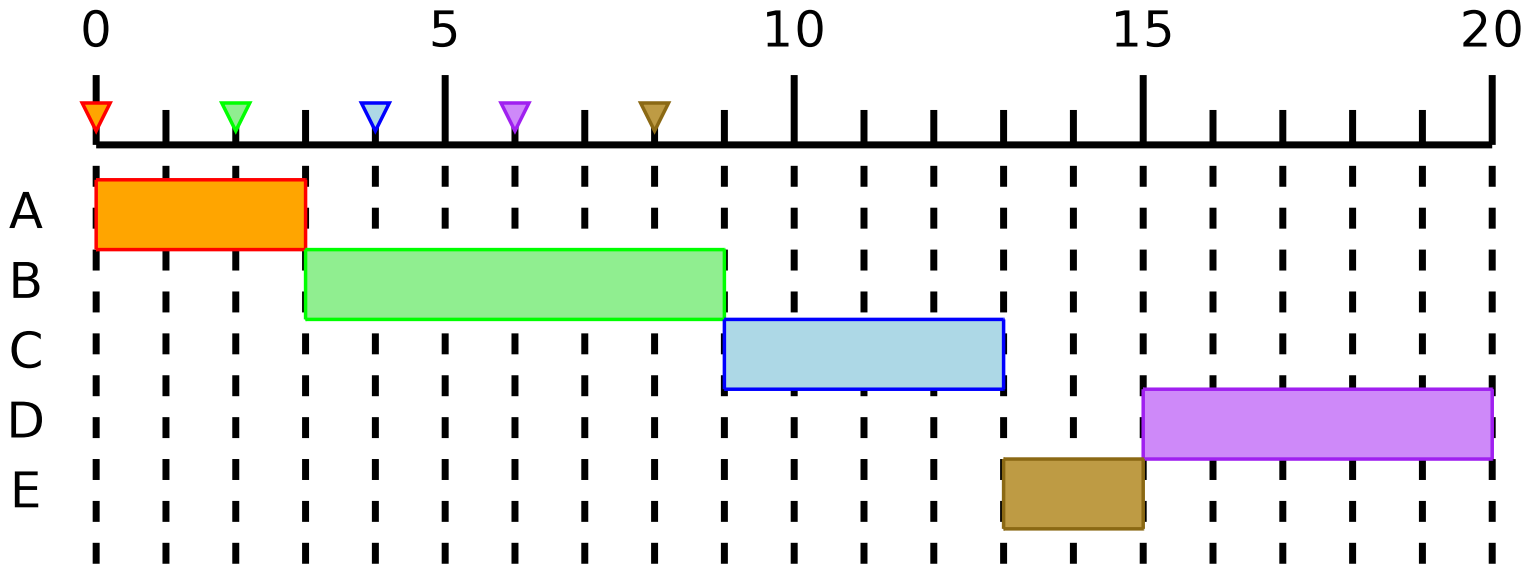


Análisis de "el (tiempo restante) más corto primero"

- Las hebras largas padecen **inanición** si existe un flujo continuo de hebras cortas.
- **Sin expulsión** no es adecuado en sistemas de tiempo compartido: las hebras acotadas por CPU tienen menos prioridad, pero aún así pueden monopolizar el procesador si no realizan E/S.
- Esta política incorpora prioridades de forma implícita: los trabajos más cortos son preferentes, así que **no es justa**.
- De alguna forma es necesario predecir el futuro:
  - ¿Cómo podemos hacerlo?
  - Algunos sistemas le preguntan al usuario: al enviar un trabajo solicitan su longitud y para evitar trampas es finalizado si incumple el plazo.
  - Incluso los usuarios más colaboradores pueden equivocarse/desconocer el tiempo de ejecución de un trabajo.

La mayor tasa de respuesta siguiente

"Highest Response Ratio Next" (HRRN) o Planificación garantizada (Tanenbaum)



Función de selección: hebra con **mayor** T_R.

Modo de decisión: **no expulsivo**

- La tasa de respuesta es una especie de "prioridad dinámica".
  $T_R = \frac{\textit{tiempo de espera} + \textit{tiempo de procesamiento}}{\textit{tiempo de procesamiento}}$
- Se favorecen los trabajos cortos sin retrasar indefinidamente a los largos porque su T_R crece con el paso del tiempo.

Planificación por lotería (1)

- Dar a cada hebra unos cuantos números de lotería.
- En cada interrupción de reloj escoger un número al azar.
- El dueño del número obtiene el procesador.
- El comportamiento del algoritmo depende de cuántos números posea cada hebra.
- ¿Cómo repartir los números?
  - Para simular primero el más corto daremos más números a las hebras más cortas.
  - Para evitar la **inanición** cada trabajo consigue **al menos un número**.

Planificación por lotería (2)

- Tiene la ventaja sobre la política de prioridad estricta de acomodarse a las cargas cambiantes.
  - Añadir o eliminar hebras afecta a todas las demás de forma proporcional, independientemente de cuántos números posea cada una.
- Las tareas pueden prestarse números entre sí (donación):
  - Un cliente pasa sus números a un servidor para que este responda rápidamente a su petición.
  - ¿Cuántos números debe poseer un servidor sin clientes?
- ¿Como implementar los números de lotería y la forma de escogerlos?

Ejemplo de planificación por lotería

- Asignaremos 10 números a los trabajos cortos y 1 a los largos.

cortos/largos	% CPU cortos	% CPU largos
1/1	90.9 %	9.1 %
0/2	–	50 %
2/0	50 %	–
10/1	9.90 %	0.99 %
1/10	50 %	5 %

- ¿Qué pasaría si hubiese muchos trabajos cortos?
  - En Unix, si la media de carga sube a 100 es difícil progresar.
  - Solución: pasar trabajos al área de intercambio.

Planificación justa

"Fair Share Scheduling" (FSS)

- En sistemas multiusuario cada usuario puede ejecutar varias tareas de forma concurrente.
- Los usuarios pueden pertenecer a grupos de usuarios y cada uno de estos grupos disponer de una cuota de uso del procesador.
- Es más fácil entender la filosofía de este planificador con un ejemplo:
  - Si hay varios grupos igualmente importantes y uno de ellos lanza más procesos que los demás la degradación del tiempo de respuesta y estancia para sus trabajos debe ser más acentuada que la de los demás grupos.

Planificación por prioridad

Función de selección: **hebra preparada** de **mayor prioridad**.

Modo de decisión: **expulsivo** (más complicado) o **no expulsivo**.

- La versión expulsiva es más compleja de programar.
- La versión no expulsiva sufre dos problemas:
  - inanición.
  - inversión de prioridad.
- La mayoría de los sistemas operativos de los ordenadores personales ofrecen algún tipo de planificación basada en prioridades con expropiación, prioridades dinámicas y alguna forma de turno circular.

Conclusiones

Resumen sobre políticas de planificación

- ¿Qué política de planificación es mejor?
- La respuesta depende de...
  - carga del sistema (extremadamente variable)
  - soporte hardware de planificadores
  - importancia relativa de los criterios: tiempo de respuesta, estancia, rendimiento, uso del procesador,...
  - método de evaluación del planificador.
- La respuesta depende de demasiados factores para poder dar una respuesta concluyente y satisfactoria.

Evaluación de las políticas de planificación

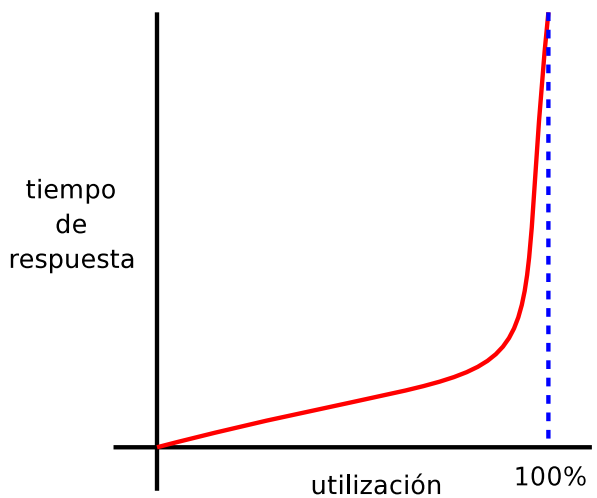
- Modelos **deterministas**:
  - Crear/recoger una carga para el sistema y probar con ella el rendimiento de cada algoritmo.
- Modelos de **colas**:
  - Igual que el método anterior pero cuando consideramos cargas de trabajo estocásticas.
- **Implementación/simulación**:
  - Construir el sistema de forma que se puedan probar los algoritmos reales sobre cargas reales o programas de prueba.

Conclusiones (1)

- Consejos contradictorios por parte de los expertos:
  - "Con prioridades puede hacerse todo".
  - "No utilizar prioridades en absoluto"
- ¿Cuando importan realmente la justicia y los detalles de los algoritmos de planificación?
  - Cuando no hay suficientes recursos para todos.

Conclusiones (2)

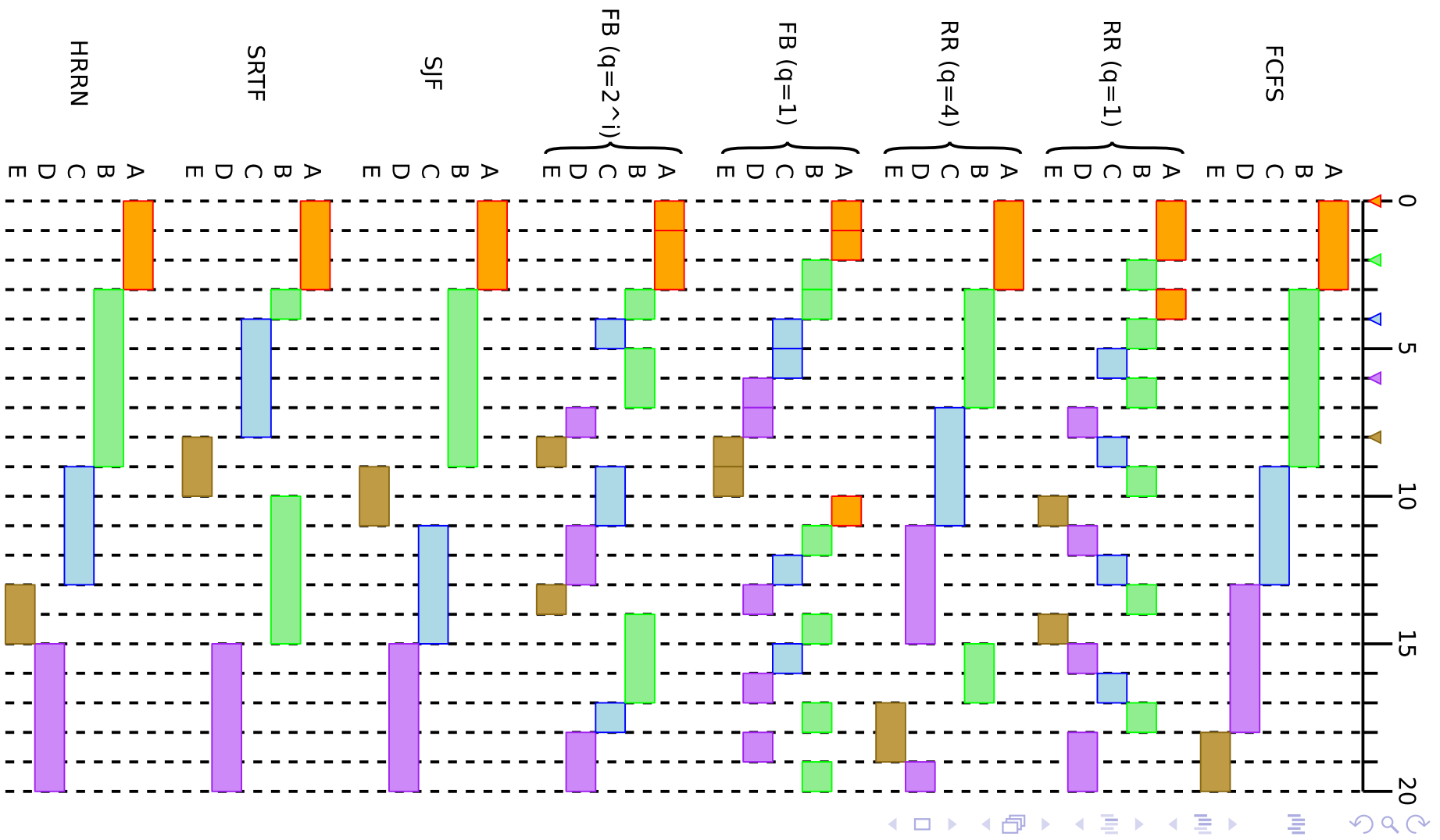
- ¿Cuando comprar un ordenador nuevo?
  - Cuando merezca la pena por lo que nos ahorramos en tiempos de respuesta y estancia.
  - A medida que nos aproximamos al 100 % de utilización de un sistema el tiempo de respuesta suele tender a infinito.
- Implicaciones interesantes de la curva de utilización:
  - La mayoría de los algoritmos de planificación funcionan bastante bien en la parte lineal de la curva y fallan después.
  - El que la utilización llegue al codo de la curva es un buen argumento para comprar un sistema nuevo.



Políticas de planificación comunes

- Los SO con aplicaciones **interactivas** suelen planificar con **expulsión**.
- Los sistemas comerciales suelen utilizar una combinación de...
  - Mecanismos de **reparto de tiempo**: expulsión por tiempo.
  - **Prioridades**: clasificación de las diferentes clases de tareas.
- Las prioridades suelen ser una combinación de...
  - Parte **estática**: tipo de tarea.
  - Parte **dinámica**: reflejando el comportamiento de la tarea y de la carga del sistema.

Políticas de planificación (Stallings)



Resumen de políticas de planificación (Stallings)

Política	Función de selección	Modo de decisión	Rendimiento	Tiempo de respuesta	Sobrecarga	Efectos sobre los procesos	Inanición
FCFS	máximo tiempo de espera	no expulsivo	no enfatizado	puede ser alto si hay mucha diferencia entre tiempos de servicio	mínima	penaliza procesos cortos o con mucha E/S	no
RR	primera preparada	expulsivo (por quantum)	puede ser pequeño si el quantum es demasiado pequeño	buen tiempo de respuesta para procesos cortos	mínimo	trato justo	no
FB	prioridad entre colas + RR/FCFS	expulsivo (por quantum)	no enfatizado	no enfatizado	puede ser alto	favorece a procesos acotados por E/S	posible
SJF	mínimo tiempo de servicio	no expulsivo	alto	buen tiempo de respuesta para procesos cortos	puede ser alta	penaliza a los procesos largos	posible
SRTF	mínimo tiempo de (servicio - empleado)	expulsivo (a la llegada)	alto	buen tiempo de respuesta	puede ser alta	penaliza a los procesos largos	posible
HRRN	máxima tasa de respuesta	no expulsivo	alto	buen tiempo de respuesta	puede ser alto	buen balance	no

Comparativa de políticas de planificación (Stallings)

	Proceso	A	B	C	D	E	Media	Varianza
	T. llegada	0	2	4	6	8		
	T. servicio	3	6	4	5	2		
FCFS	T. inicio	0	3	9	13	18	8,60	7,30
	T. respuesta	0	1	5	7	10	4,60	4,16
	T. salida	3	9	13	18	20	12,60	6,88
	T. estancia	3	7	9	12	12	8,60	3,78
	estancia/servicio	1	1	2	2	6	2,56	2,02
RR (q=1)	T. inicio	0	2	5	7	10	4,80	3,96
	T. respuesta	0	0	1	1	2	0,80	0,84
	T. salida	4	18	17	20	15	14,80	6,30
	T. estancia	4	16	13	14	7	10,80	5,07
	estancia/servicio	1	3	3	3	4	2,71	0,84
RR (q=4)	T. inicio	0	3	7	11	17	7,60	6,69
	T. respuesta	0	1	3	5	9	3,60	3,58
	T. salida	3	17	11	20	19	14,00	7,07
	T. estancia	3	15	7	14	11	10,00	5,00
	estancia/servicio	1	3	2	3	6	2,71	1,71
FB (q=1)	T. inicio	0	2	4	6	8	4,00	3,16
	T. respuesta	0	0	0	0	0	0,00	0,00
	T. salida	4	18	15	20	16	14,60	6,23
	T. estancia	4	16	11	14	8	10,60	4,77
	estancia/servicio	1	3	3	3	4	2,71	0,94
FB (q=2^i)	T. inicio	0	3	4	7	8	4,40	3,21
	T. respuesta	0	1	0	1	0	0,40	0,55
	T. salida	3	17	18	20	14	14,40	6,73
	T. estancia	3	15	14	14	6	10,40	5,50
	estancia/servicio	1	3	4	3	3	2,56	0,94
SJF	T. inicio	0	3	11	15	9	7,60	6,07
	T. respuesta	0	1	7	9	1	3,60	4,10
	T. salida	3	9	15	20	11	11,60	6,39
	T. estancia	3	7	11	14	3	7,60	4,88
	estancia/servicio	1	1	3	3	2	1,84	0,87
SRTF	T. inicio	0	3	4	15	9	6,20	5,89
	T. respuesta	0	1	0	9	1	2,20	3,83
	T. salida	3	15	8	20	10	11,20	6,53
	T. estancia	3	13	4	14	2	7,20	5,81
	estancia/servicio	1	2	1	3	1	1,59	0,84
HRRN	T. inicio	0	3	9	15	13	8,00	6,40
	T. respuesta	0	1	5	9	5	4,00	3,61
	T. salida	3	9	13	20	15	12,00	6,40
	T. estancia	3	7	9	14	7	8,00	4,00
	estancia/servicio	1	1	2	3	4	2,14	1,07